

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

COMMUNICATIONS SYSTEM FOR ELECTRONIC COMMERCE

Patent Number: WO9957864
Publication date: 1999-11-11
Inventor(s): JONGLEZ MATTHIEU JEAN NADY (GB); TOWNDROW STEPHEN PETER (GB)
Applicant(s):: BRITISH TELECOMM (GB); JONGLEZ MATTHIEU JEAN NADY (GB); TOWNDROW STEPHEN PETER (GB)
Requested Patent: WO9957864
Application Number: WO1999GB01332 19990429
Priority Number (s): EP19980303539 19980506
IPC Classification: H04L29/06 ; G06F17/60 ; G07F7/10
EC Classification: G06F17/60B, G07F7/10F6, H04L29/06
Equivalents: AU3720599

Abstract

In a communications network used for electronic commerce, a merchant server sends to a customer terminal data on items available for purchase. The data is displayed at the customer terminal, for example using a web browser. The customer selects items to be purchased and a cumulative list of the selected items (the "shopping basket") is generated at the customer terminal automatically and independently of the merchant server. The cumulative list may be generated by a program encapsulated, for example as a Javascript (Trade Mark) function, in the data downloaded from the merchant server.

Data supplied from the esp@cenet database - I2

Description

COMMUNICATIONS SYSTEM FOR ELECTRONIC COMMERCE BACKGROUND TO THE INVENTION

The present invention relates to a communications system, and in particular to a system designed to support electronic commerce.

The growth of the internet and world wide web has made electronic commerce an important field, and much effort has been directed to designing systems which facilitate on-line trading. However, while internet retail sites for occasional and relatively high-value purchases, such as on-line bookstores, have achieved some success, the variability in the quality of service offered by the internet, and in particular the slowness of connections at times of peak usage, has inhibited the more widespread acceptance of on-line retailing by customers.

SUMMARY OF THE INVENTION

A method of operating a communications system comprising a merchant server, at least one customer terminal, and a communications network linking the merchant server to the customer terminal, the method comprising:

- a) dynamically generating at a database remote from the customer terminal a set of data identifying a plurality of items available for purchase and communicating to the customer terminal from the merchant server the said set of data identifying a plurality of items available for purchase;
- b) displaying the said data at the customer terminal;
- c) registering at the customer terminal a selection made by the customer of one or more of the plurality of items;
- d) generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by
 - (i) instantiating a first data array
 - (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;
- e) updating the said list at the customer terminal in response to any subsequent selection by the user of further items;
- f) displaying the said list at the customer terminal; and
- g) in response to a further input from the user, communicating via the communications network to the merchant server a purchase order including data from the list.

Conventionally, on-line retail sites have provided the customer with a cumulative list of items selected for purchase. The merchant server records each selection and returns a modified list to the customer terminal for display. For obvious reasons, this list is termed the "shopping basket". The present invention dynamically generates a set of data from a database, transfers this data to the customer, and then shifts the step of generating and displaying the shopping basket entirely to the customer terminal, so that it can be carried out without requiring further exchange of data with the merchant server. This dramatically increases the perceived speed of operation for the user, and reduces the variability of the speed of response with changes in the communications network.

The customer terminal may be a personal computer connected to the internet, a mobile terminal, a multimedia kiosk such as BT's Touchpoint (Trade Mark) or any other suitable device.

Preferably step (a) includes communicating a web page including the said data to the customer terminal and step (b) is carried out by a web client application running at the customer terminal. Preferably the method includes communicating with the said data in step (a) a program to be executed at the customer terminal, which program carries out at least step (d).

It is found to be particularly advantageous to download with the data the program which generates and displays the shopping basket at the customer terminal. For example, the program might take the form of Javascript contained within an HTML page (JavaScript is a Trade Mark of Sun Microsystems). Then although the data processing for the shopping basket takes place entirely at the customer terminal, the

method by which the data is processed is nonetheless determined by the merchant server platform. Then, for example, if it is necessary to change the format in which the data is communicated to the merchant when the order is confirmed, this change can be effected simply by modifying appropriately the program.

Preferably at least part of the data defining the cumulative list of items is stored by the client terminal as a cookie. The cookie need not contain all of the data elements in the shopping basket data array, but may only contain as much data as is needed to reconstitute that array after, for example, a system crash, or the user logging out. For example, the cookie might store a user ID and the identities and quantities of the selected items. The use of a cookie in this manner gives the method of the invention as much, or a greater degree of robustness as conventional server-side shopping baskets, while still avoiding the disadvantages of requiring frequent communication with a server-side database.

Preferably the method further comprises:

(h) calculating at the customer terminal from data elements in the first data array a total price for purchase items selected by the user, and displaying the said total price at the customer terminal.

In this case, step (d) of the method may further comprise:

(iii) instantiating a second data array and, when a special offer purchase item is selected, populating both the first and second data arrays with data elements from the said set of data corresponding to the selected purchase item, the data elements in the second array including, for each selected special offer purchase item, data determining a modification to the price of the said purchase item; and in step (h) the total price is calculated from data elements in both the first and second data arrays.

The inventors have found that the flexibility in operation and maintenance of the electronic commerce system is significantly enhanced by the use on the customer terminal of a second data array specifically for those purchase items to which special offers apply. This allows new special offers to be implemented without extensive changes to the core data, or to the basic functions used, for example, to calculate a price for the shopping basket. It also enables the implementation of complex special offer schemes, for example where the customer is invited to purchase one item and then to receive associated items free of charge.

According to a second aspect of the present invention, there is provided a communications system comprising;

- a) means for dynamically generating at a database remote from the customer terminal a set of data identifying a plurality of items available for purchase and communicating to the customer terminal from the merchant server the said set of data identifying a plurality of items available for purchase;
- b) means for displaying the said data graphically at the customer terminal;
- c) means for registering at the customer terminal a selection made by the customer of one or more of the plurality of items; (d) means for generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by
- (i) instantiating a first data array
- (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;
- e) means for updating the said list at the customer terminal in response to any subsequent selection by the user of further items;
- f) means responsive to a further input from the user, for communicating via a communications network to the merchant server a purchase order including data from the list.

According to a third aspect of the present invention, there is provided a communications system comprising

- a) a merchant server comprising:
 - a database programmed with data identifying a plurality of items available for purchase;
 - a data output arranged to output a set of data dynamically generated from the database for display at a remote terminal;
 - a signal input for receiving purchase orders returned from a customer terminal;
- b) a communications network connected to the merchant server;
- c) at least one customer terminal connected to the communications network, the customer terminal comprising:
 - means for displaying data received from the merchant server;
 - means for registering a selection made by the user of one or more of the plurality of items;

means for generating automatically and independently of the merchant server a cumulative list of items selected by the user by generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by

(i) instantiating a first data array (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;

means responsive to a further input from the user for communicating via the communications network to the merchant server a purchase order including data from the said cumulative list of items.

The invention also comprises merchant servers and customer terminals adapted for use in accordance with the preceding aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

Systems embodying the present invention, will now be described in further detail, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a schematic of a network embodying the invention;
 Figure 2 shows a web page displayed at a customer terminal;
 Figure 3 shows a first stage in loading data on the vendor server;
 Figure 4 shows a second stage in loading data on the vendor server;
 Figure 5 shows a third stage in loading data on the vendor server;
 Figure 6 shows a report returned to the vendor server during the third stage; and
 Figure 7 is a flow diagram for the process of loading data.

DESCRIPTION OF EXAMPLES

A customer terminal 1, which in the present example is a personal computer running a web browser such as Microsoft Corporation's Internet Explorer (Trade Mark), is connected to a data network 2. In the present example the data network is a packet network using TCP/IP technology. A vendor server 3 is connected to the network 2. This is a computing platform running, e. g., commercially available software, such as Oracle's WebServer (Trade Mark). The vendor server serves to the customer terminal HTTP (hypertext transfer protocol) pages generated from data stored on the vendor server in a database 4. In this example, the vendor server is operated on behalf of the vendor by another party, such as a network operator. The vendor has their own local server 300, which uploads data to the vendor server and downloads customer orders from the vendor server. The data connection between the local server 300 and the vendor server may be made via the same network 2 as that used for the connection between the vendor server 3 and the customer terminal.

Figure 2 shows an example of a web page served by the vendor server 3 to a customer terminal 1. In this example, the vendor is a food retailer operating an on-line supermarket. A left hand frame 21 lists different categories of goods, for example frozen goods, hot beverages, laundry and cleaning. Each item in this list is a link which, when selected by the customer, returns to the central frame 22, a list of further sub-categories, such as frozen vegetables, ready-made meals, etc. for the "frozen goods" category. When the user selects one of these sub-categories by clicking on the relevant link, a list of the products in question is returned to the central frame 22. Each time a category or sub-category link is selected by the user, this is communicated by the web browser to the vendor server. At the vendor server, a PL/SQL query is run to address data in the database 4 in order to return to the customer, e. g., the list of sub-categories or the list of products. In this way, the database 4 dynamically generates a set of data, which is sent to the customer in an HTML page. PL/SQL is an enhanced version of SQL (standard query language) available commercially from Oracle and designed to provide a procedural language for programs to query relational databases. The results of the PL/SQL query are stored in a Javascript (Trade Mark) function encapsulated in a dynamically generated HTML page displayed on the customer terminal.

In the example illustrated in Figure 2, the data displayed in the central frame 22 is a list of goods (in this case frozen vegetables) with their prices and buttons 220 which may be selected by the user to add items from the list of goods to a shopping basket, that is a cumulative list of goods which have been selected for purchase. In response to the selection by the user of the button 220, the Javascript program encapsulated in the HTML page adds the corresponding data item to a shopping basket array variable, which stores data on all the items selected for purchase in the course of transaction. If there is a special offer associated with the selected goods, then the data item is also added to a special offer Javascript array variable which stores data on all selected items to which

special offers apply. In addition, the shopping basket as updated with the selected data item is stored as a cookie, that is as a data item which is stored in a predefined format and which is returned by the web client to the server when the client next transmits an HTTP request to the server. Cookies are designed to allow persistent state information to be stored and typically will be saved by the web browser in a predetermined directory of a mass storage device at the customer terminal, such as the hard disk of a PC. The Javascript variables are held in RAM for duration of the transaction.

After running the functions add to-basket and/or add to special offer, the Javascript program running at the customer terminal uses a further Javascript function, draw~basket, to generate a display of the contents of the shopping basket. This is output to a third frame 23 at the bottom of the web page. Each item in the basket has beside buttons for deleting or amending the item. Selecting the buttons triggers further Java functions running on the customer terminal to amend the data arrays accordingly. The add to basket, add to special-offer and draw-basket functions are defined in the Javascript source code listing set out in the Table below.

Tables 2 and 3 show examples of, respectively, a first data array corresponding to the main shopping basket array, and a second data array corresponding to special offer purchase items. In operation, a Java script program running on the customer terminal initiates these two variable length data arrays, and then populates these arrays with data elements from the set of data contained in the dynamically generated HTML page, as the corresponding purchase items are selected. As shown in Table 2, the data elements stored in the first array include an ID for each selected item, a variable indicating the quantity of each items selected, the price of each item, a textual description of each item, a status flag, and a flag string. The flag string includes a number of fields a, b, c..."SO" is written into one of these fields in the case of purchase items to which a special offer applies. For such purchase item, corresponding data elements are also written into the second data array, as illustrated in Table 3. This includes a description of the relevant special offer. In the example shown, the offer is "buy chicken, get free pie, peas and potatoes". A trigger quantity is specified also. In this example this has the value 1 for the chicken, since any purchase of the chicken triggers the offer. However, for other special offer schemes, for example for a scheme of the type "buy 2 get 1 free" the trigger quantity may have a higher value and the special offer scheme will not be applied when the quantity purchased is below the quantity trigger value. Purchase items are specified in the related product ID field, when the special offer is dependent upon some other product having been purchased. The array also includes data elements identifying any related group, the item's own group, and the discount applied to the item. In the case of the illustrated special offer, a zero discount is applied to the chicken.

However, for the related items in the special offer array, the pie, peas and roast potatoes, the discount field has a discount of 1, that is a 100% discount is applied so that the item is free to the purchaser. Subsequently, a total price calculation function included in the Java script downloaded to the customer terminal steps through the first array and for items with no special offer flag string set, calculates an addition to the cumulative price simply by multiplying the item quantity and the unit price. For other items, where the special offer flag is set, then as well as reading the price from the first array, the discount is read from the second array and the appropriate discount applied in calculating the total cost of the shopping basket.

More than one type of special offer scheme may be used. For example, in addition to the "buy chicken, get free pie, peas and potatoes" scheme illustrated in Tables 2 and 3, a "buy two get one free" scheme may apply to others.

The customer may subsequently select other categories from the left hand frame 21, and add other goods to the shopping basket. When all the selections have been made the customer selects an "order" button. This causes an HTTP request to be returned to the vendor server. The HTTP request includes the cookie containing the shopping basket data. This data is stored at the vendor server. Once the order has been placed, the contents of the cookie and the Javascript variables at the customer terminal are cleared. In response to the HTTP request the vendor server may return to the customer terminal a new HTTP page containing a confirmation that the order has been successfully placed, and a menu giving options for delivery slots for the ordered goods. The selection of delivery slot by the user is returned to the server and stored together with the order. The order as a whole is later downloaded to the vendor local server 300, for processing of the order, resulting in the subsequent dispatch of the ordered goods from a warehouse to a delivery address specified by the customer.

Figures 3 to 5 illustrate the steps by which the vendor uploads data from the local server 300 to the vendor server 3. The local server first sends an HTTP request the vendor server. A CGI script at the vendor server generates dynamically an HTTP page of the form shown in Figure 3. The operator of the local server 300 enters in the dialogue box 301 the name of a comma delimited . csv file 302 which contains product names and prices. This file may be created, for example, within a spreadsheet. The vendor server 3 then returns to the local server 300 an HTTP page of the form shown in Figure 4. This asks the vendor to identify the type of data in the . csv file, for example as products, prices or categories. By clicking on the appropriate link, a request is returned to the server 3 which causes the server 3 to upload the data from the . csv file into a staging table. The staging table is a temporary mirror for part of the database. When the data has been uploaded, the server returns the page shown in Figure 4. The user is requested to click on a link to initiate checking of the validity of the uploaded data. A PL/SQL program running on the server 3 then checks the validity of the data against predetermined criteria-for example the check may identify any categories for which there are no products, or any products for which there are no price data. As shown in Figure 5, the vendor server 3 returns to the local server 300 details of any inconsistencies, so that the vendor can amend the data in their own tables and then upload the corrected data as an amended . csv file. When the integrity of the data has been confirmed, the vendor may click on link 501 in order to preview the data as it will be displayed to a customer. The server then dynamically generates a web page like that shown in Figure 2, using the same HTTP, Javascript and PL/SQL code, but taking the data from the staging table rather than from the database 5. Once the vendor is happy with the data, and the consistency of the data has been checked, then by selecting the upload link 502, the vendor causes the data to be transferred from the staging tables to the database 5. The process of loading data described above is illustrated in the flow diagram of Figure 7.

TABLE

Javascript (TM) shopping basket source code //Add the specified item into the array and the cookie // functionaddtobasket (pcode, pqty, pname, pprice, poffer)

```
{
var exist =-1;
var vbasket = get-cookie ('SHOPPING-BASKET');
var n basket = '&num;';
var property;
var v~product;
var v~qty;
var new~quantity = 0;
var today = new Date ();
p~qty=Math. round (parseFloat (p~qty) * 100)/100;
& & !(p~offer & ~unit-lb~)if(!(p~offer & ~unit-kg~)
p~qty = Math. ceil (p~qty);
if (v~basket. length > 0) {
v~basket.substring(1,v~basket.length);v~basket= if(0)#
vproperty = v~basket. substring (0, v basket. length);
else
v~property = vbasket. substring (0, v~basket. indexOf ('&num;&num;PDT&num;&num;'));
vbasket = vbasket. substring (v~property. length, v basket. length); nbasket += vproperty;
for (var counter=0; counter < basket. length; counter++)
if (basket [counter]. productcode==pcode) exist=counter;
}
if (exist ==-1)
{
basket [basket. length] = new definebasketcontent ();
basket [basket. length]. product code=p~code;
basket [basket. length]. quantity = p~qty;
basket [basket. length]. name = p~name;
basket [basket. length]. price = p~price;
basket [basket. length]. offercode = poffer;
basket [basket. length]. status = 1 ;
basket. length++; new~quantity=p~qty;
}
else
```

```

{
basket [exist]. quantity+=p~qty;
new~quantity=basket [exist]. quantity;
texist=1; <#s>
while (v~basket. length > 7) {
v~basket = v~basket. substring (7, v~basket. length);
v~product = v~basket.substring(0,v~basket.indexOf('&num;&num;QTY&num;&num;'));
vbasket = vbasket. substring (v~product. length, v~basket. length);
vbasket = vbasket. substring (7, v~basket. length);
if (v~basket.indexOf('&num;&num;PDT&num;&num;') > 0)
v~qty = v~basket.substring(0,v~basket.indexOf('&num;&num;PDT&num;&num;'));
}
else
{
v ~ q t y = v basket. substring (0, v basket. length);
} vbasket = v~basket. substring (v~qty. length, v~basket. length);
if (v~product == p~code)
n~basket+='&num;&num;PDT&num;&num;'+p~code+'&num;&num;QTY&num;&num;'+new~quantity;
exist=1;
}
else
n~basket += '&num; &num;PDT&num;&num;'+v~product+'&num;&num;QTY&num;&num;'+v~qty;
if (exist==1)
{
n~basket+='&num;&num;PDT&num;&num;'+p~code+'&num;&num;QTY&num;&num;'+new~quantity;
}
set~cookie('SHOPPING-BASKET,n~basket,new Date(today.getYear(),
12)?0:(today.getMonth()+1)),((today.getMonth()+1== ((today.getDate() > = 28)? 28: today. getDate ()))));
}
parent. body. rbottom. location='/shop/shopping~basket. html' ; function addtospecial ! offer (pcode,
ocode, disc, desc, pck) {
var exist = 0;
for (var counter=0; counter < special~offer. length; <#s> counter++)
{
if ( (special offer [counter]. productcode==pcode)
& & exist=1;
if (exist == 0)
{
newdefine~special~offer~content();special~offer[special~offer.length]=
special~offer [special offer. length]. product code = p code;
special~offer[special~offer.length].offer~code = ocode;
special~offer[special~offer.length]. discount = disc;
special offer [special offer. length]. desc = desc;
special~off[special~offer.length]. pck = pck;
special~offer.length++; } //Just set the cookie on the client side // function set-cookie (name, value,
expirydate) {
document. cookie = name +"="+ value
+ ((expiry date == null) ? "" : ("; expires=" + expiry~date.toGMTString()))+ "path=/shop/";
} // //Just get the specified cookie and treat it on the client side // function get~cookie(name)
{
var seek = name+ "=";
if (document. cookie. length > 0)
{ offset = document. cookie. indexOf (seek) ;
if (offset != -1)
{ // if the cookie exists
offset += seek. length;//starting point
end = document. cookie. indexOf (";", offset);//ending point
if (end ==-1) end = document. cookie. length;
return document.cookie.substring(offset, end);
}
}
}
}

```

```

} //Just print a line of the shopping basket, formatting the data // function draw~basket~line(p~code,
o~code, item, price, qty, total, status)
{ printf ( ' < tr > < td bgcolor=navajowhite align=right > < a name=line'+p~code+' > < /a > '
+ ( (o code & special offer)? ( (status == 1)? (' < a href="javascript: parent. parent. support. show offer ('
+ pcode+'\') "OnMouseOver="window. status='\Show details of the special offer'; <#s> return true"
+ ' OnMouseOut="window.status='\';return true"##;img src='+server~name
+ /images/arrowright. gifborder=0 > < /a > '); (' < a href="javascript: parent. parent. support. hideoffer ('
+ p~code+'\') " OnMouseOver="window. status='\Hide details of the special offer'; return true"
+ 'OnMouseOut="window. status='\';return true" > < img src='+server name
+ /images/arrow down. gif border=0 > < /a > ');' & nbsp;')
+ ' < /td > < td bgcolor=navajowhite > '
+ ' < a
href="" +server~name+ /merchant/owa/categories.show~product~details?shop~code='+p~code+""
+ ' OnMouseOver="window.status='\Display details for '+item+'\';return true"
+ ' OnMouseOut="window.status='\'; <#s> return true" target=main#+item
+ ' < /a > < /td > < td align=right bgcolor=navajowhite > '
+ ( (navigator. appName=="Netscape")? ((price < 1) && (0 < price))? ('0'+price): price):
price) + ( (Math. ceil (price) == price)? ('. 0') :")
+ ( (Math. ceil (price* 10)/10 == price ) ? ('0') :")
kg':((o~code & ~unit~lb~)?'perlb':'each'))+((o~code & ~unit~kg~)?'per '+#/td#td align=right
bgcolor=navajowhite#+qty+#/td#
+ ' < td align=right bgcolor=navajowhite > '
+ ( (navigator. appName=="Netscape")? ((total < 1) && (0 < total))? ('0'+total): total): <#s> total )
+ ( (Math. ceil(total) == total) ? ('. 0') :")
+ ( (Math. ceil (total* 10)/10 == total) ? ('0') :") +#/td#td align=center bgcolor=navajowhite#'
+ ' < ahref="javascript: parent. parent. support. remove~item('\'+p~code+'\';"
+ 'OnMouseOver="window. status='\Remove item... \';return true"
+ 'OnMouseOut="window.status='\';return true"#
+ ' < img src="" +server~name+ /images/rect. gif' border=0 > < /a > < /td > < td align=center
bgcolor=navajowhite > '
+ ' < ahref="/shop/amendbasket? code='+pcode+""
+ 'OnMouseOver="window. status='\Amend item... \'; <#s> return true"
+ 'OnMouseOut="window.status='\';return true"# +'#img src="" +server~name+ /images/rect.gif' border=0
##/a##/td##/tr#);
} // // function draw~special~offer~line(p~code)
{
for (var counter=0 ; counter < special~offer. length; <#s> counter++)
{
if (special offer [counter]. productcode==pcode) printf ( ' < tr > < td > < /td > < td bgcolor=navajowhite
align=right > < b > '+special~offer [counter]. desc
+ ' < /b > < /td > < td colspan=5 > < /td > < /tr > ');
} // //Just draw the line containing the total amount of the shopping basket //
function draw basket total (total, nb~item) {
{ printf ( ' < tr > < td colspan=4 align=right bgcolor=&num;DFDFFF > < b > order total amount : ' +
((shipping~comment.length#0) ? '#i#('+shipping~comment+')#/# : ') + '#/b##/td##td align=right
bgcolor=&num;DFDFFF##b## & nbsp;@
+ ( (navigator. appName=="Netscape")? ((total < 1) && (0 < total))? ('0'+total): total) : total
)
+ ( (Math. ceil (total) == total)? ('. 0') :")
+ ( (Math. ceil (total* 10)/10 == total ) ? ('0') :")
+ ' < /b > < /td > < td colspan=2 bgcolor=&num;DFDFFF align=left > < b > < i > ('+nb item
}
+ 'item'+ ( (nb~item > 1) ? 's:')+' )#/b##/i# #/td##/tr##/table#) ;
//
//Just print the closing html tags in the basket frame
//
function draw~document~trailer()
printf ( ' < /center > < /body > < /html > ');
}
//
//Just draw the title line of the amend screen

```



```

numero=sp~index;
}
subtotal = Math. round (basket [counter]. price*
(Math. floor ( basket [counter]. quantity/special offer [numero]. pck)
* special offer [numero]. discount +
(basket [counter]. quantity % special offer [numero]. pck)
) *100)/100;
}
if (counter > = limit)
draw~basket~line(basket[counter].product~code,basket[counter].offer~code,
basket[counter].name, price, basket[counter].quantity,
sub total, basket [counter]. status);
if ( (basket [counter]. offer code & special offer) && (basket [counter]. status==2))
draw special offer line (basket [counter]. product code); total amount+=sub total;
~unit~kg~);if((basket[counter].offer~code &
(basket[counter].offer~code & ~unit~lb~))
{
nb~item += 1;
}
else
nbitem += basket [counter]. quantity;
draw basket total (Math. round (total~amount*100)/100, nb~item) ;
draw~document~trialer(); Math.round(total~amount*100)/100;basket~current~total=
}

```

Table 2

| Item/I. | D. | Quantity | Price | Description | Status | Flag |
|---------------|----|----------|--------|--------------------|--------|------|
| Chicken/1 | 2 | "text" | O. K | (a, b, c, so) | | |
| Pie/2 | 1 | "text" | O. K | (a, b, c, so) | | |
| Peas /3 | 1 | #.6 | "text" | O.K. (a, b, c, so) | | |
| Roast Pots./4 | 1 | #.6 | "text" | O.K. (a, b, c, so) | | |

Table3

| Item/I. | D. | Description | Quantity | Related | Related | Group | Discount | Special Offer |
|----------------|------------|-------------|----------|---------|---------|-------|----------|---------------|
| Trigger | Product Id | Group | Scheme | | | | | |
| "buy1-AA0Z | Chicken/1 | | | | | | | |
| chicken, get | | | | | | | | |
| free pie, peas | | | | | | | | |
| and pots." | | | | | | | | |
| Pie/2 | As above | 0 | A | Z | | | | |
| Peas/3 | As above | 0 | A | Z | | | | |
| Roast Pots/4 | As above | 0 | 1-A | 1 | Z | | | |

Data supplied from the esp@cenet database - I2

Claims

CLAIMS 1. A method of operating a communications system comprising a merchant server, at least one customer terminal, and a communications network linking the merchant server to the customer terminal, the method comprising:

- a) dynamically generating at a database remote from the customer terminal a set of data identifying a plurality of items available for purchase and communicating to the customer terminal from the merchant server the said set of data identifying a plurality of items available for purchase;
- b) displaying the said data at the customer terminal;
- c) registering at the customer terminal a selection made by the customer of one or more of the plurality of items;
- d) generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by
 - (i) instantiating a first data array
 - (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;
- e) updating the said list at the customer terminal in response to any subsequent selection by the user of further items;
- f) displaying the said list at the customer terminal; and
- g) in response to a further input from the user, communicating via the communications network to the merchant server a purchase order including data from the list.

2. A method according to claim 1, in which step (a) includes communicating a web page including the said data to the customer terminal and step (b) is carried out by a web client application running at the customer terminal.

3. A method according to claim 1 or 2, including communicating with the said data in step (a) a program to be executed at the customer terminal, which program carries out at least step (d).

4. A method according to claim 3 when dependent on claim 2, in which at least part of the data defining the cumulative list of items is stored by the client terminal as a cookie.

5. A method according to any one of the preceding claims, further comprising;
(h) calculating at the customer terminal from data elements in the first data array a total price for purchase items selected by the user and displaying the said total price at the customer terminal.

6. A method according claim 5, in which step (d) further comprises:
iii) instantiating a second data array and, when a special offer purchase item is selected, populating both the first and second data arrays with data elements from the said set of data corresponding to the selected purchase item, the data elements in the second array including, for each selected special offer purchase item, data determining a modification to the price of the said purchase item;
and, in which in step (h), the total price is calculated from data elements in both the first and second data arrays.

7. A method according to any one of the preceding claims, further comprising a step of uploading to the merchant server from a source remote from the merchant server data identifying a plurality of items available for purchase, and in which the said step of uploading includes storing data uploaded to the vendor server in a staging table, checking the validity of the said data, and only transferring the said data from the staging table to the database used to generate the display in step (b) depending on the result of the said checking of the validity.

8. A method according to claim 7, including, as part of the step of uploading data to the vendor server, a step of generating from data held in the staging table a preview of a display corresponding to the display of step (b).

9. A method according to any one of claims 5 to 7, in which the step of uploading includes requesting a web page from the vendor server, and indicating in data entry means provided in the said web page the

location at the remote source of the data to be uploaded.

10. A communications system comprising;

- a) means for dynamically generating at a database remote from the customer terminal a set of data identifying a plurality of items available for purchase and communicating to the customer terminal from the merchant server the said set of data identifying a plurality of items available for purchase;
- b) means for displaying the said data graphically at the customer terminal;
- c) means for registering at the customer terminal a selection made by the customer of one or more of the plurality of items;
- (d) means for generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by
 - (i) instantiating a first data array
 - (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;
- e) means for updating the said list at the customer terminal in response to any subsequent selection by the user of further items;
- f) means responsive to a further input from the user, for communicating via a communications network to the merchant server a purchase order including data from the list.

11. A communications system comprising

- a) a merchant server comprising:
 - a database programmed with data identifying a plurality of items available for purchase;
 - a data output arranged to output a set of data dynamically generated from the database for display at a remote terminal;
 - a signal input for receiving purchase orders returned from a customer terminal;
- b) a communications network connected to the merchant server;
- c) at least one customer terminal connected to the communications network, the customer terminal comprising:
 - means for displaying data received from the merchant server;
 - means for registering a selection made by the user of one or more of the plurality of items;
 - means for generating automatically and independently of the merchant server a cumulative list of items selected by the user by generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by
 - (i) instantiating a first data array
 - (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;
 - means responsive to a further input from the user for communicating via the communications network to the merchant server a purchase order including data from the said cumulative list of items.

12. A system according to claim 10 or 11, in which the means for generating include a program communicated from the merchant server to the customer terminal with the said data identifying a plurality of items.

13. A system according to any one of claims 10 to 12, in which the communications network is a network supporting a packet-based internetworking protocol.

14. A system according to any one of claims 10 to 13, in which the merchant server includes a web server arranged to serve web pages to the or each customer terminal.

15. A merchant server arranged for use in a method according to any one of claims 1 to 9 or in a system according to any one of claims 10 to 14.

16. A customer terminal arranged for use in a method according to any one of claims 1 to 9 or in a system according to any one of claims 10 to 14.

17. A merchant server for use in a method according to any one of claims 1 to 9, the merchant server comprising:

- a store programmed with data identifying a plurality of items available for purchase;
- a data output arranged to output data from the data store for display at a remote terminal;

a signal input for receiving purchase orders returned from a remote terminal.

18. A merchant server according to claim 17, in which the store further comprises a client-side program, which program is, in use, communicated to the customer terminal, the client-side program being arranged to generate automatically and independently of the merchant server, for display at the customer terminal, a cumulative list of items selected by the user by generating at the customer terminal automatically and independently of the merchant server a cumulative list of items selected by the user by

(i) instantiating a first data array

(ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or each item selected by the user;

19. A customer terminal for use in a method according to any one of claims 1 to 9, comprising:

means for receiving and displaying data identifying a plurality of items available for purchase;

means for registering a selection made by the user of one or more of the plurality of items; and

means for generating automatically and independently of the merchant server a cumulative list of items selected by the user by:

(i) instantiating a first data array (ii) populating the first data array with data elements from the said set of data, the said elements corresponding to the or

each item selected by the user; 20. A customer terminal according to claim 19, including a region of memory and in which the first data array is stored in the region of memory.

21. A customer terminal according to claim 20, including a second data array stored in the region of memory, and in which both the first and second data arrays are populated with data elements corresponding to selected special offer purchase items, the data elements in the second array including, for each selected special offer purchase item data determining a modification to the price of the said special offer purchase item, and in which the customer terminal further comprises price calculation means arranged to calculate a total price for purchase items selected by the user from data elements in both the first and second data arrays, and to display the said total price.

Data supplied from the esp@cenet database - I2

Fig. 1.

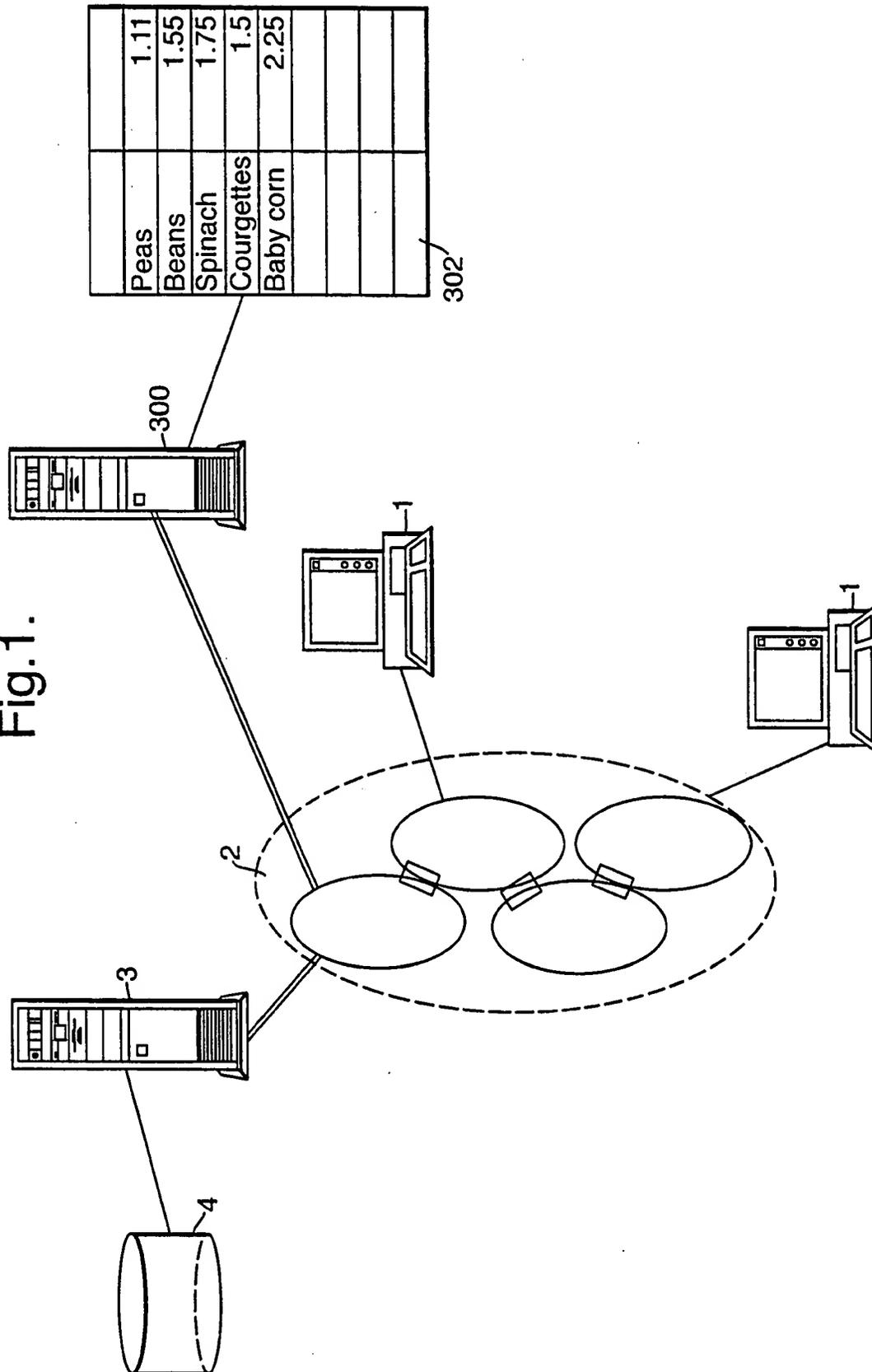


Fig.2.

The Shop - Netscape
<http://132.146.75.157:7878/shop/index.html>

Shopping Basket Product Search Order History

Large Normal Off Search now Display now

Frozen Vegetables

22

Product Details

| | | | |
|------------------------------------|------|--------|---------------|
| Frozen Baby Carrots 908g | each | £ 1.49 | add to basket |
| Frozen Broccoli Spears 908g (BOND) | each | £ 2.09 | add to basket |
| Frozen Brussels Sprouts 908g | each | £ 1.75 | add to basket |
| Frozen Chopped Spinach 908g | each | £ 1.95 | add to basket |
| Frozen Corn on the Cob 2pk | each | £ 0.78 | add to basket |
| Frozen Leaf Spinach 908g | each | £ 1.95 | add to basket |
| Frozen Peas 454g (Birds Eye) | each | £ 1.15 | add to basket |
| Frozen Peas 908g (Birds Eye) | each | £ 2.11 | add to basket |

23

| Special Offer | Item | Price | Quantity | Total | Remove | Amend |
|---------------|-------------------------------|-------|----------|-------|--------|-------|
| | Spiced Steak Burgers 6 - 8oz | 2.38 | 1 | 2.38 | ■ | ■ |
| | Beef Kebabs (Summer only) | 0.00 | 1 | 0.00 | ■ | ■ |
| | HD Choc Chip Ice Cream | 3.60 | 1 | 3.60 | ■ | ■ |
| | HD Belgian Chocolate | 3.60 | 1 | 3.60 | ■ | ■ |
| | Lamb Vindaloo (Mrs Gill) | 2.50 | 1 | 2.50 | ■ | ■ |
| | Pindles Sour Crm n Onion 190g | 1.45 | 1 | 1.45 | ■ | ■ |
| | Reach Nit Rate Res 22.5g ROX | 1.69 | 1 | 1.69 | ■ | ■ |

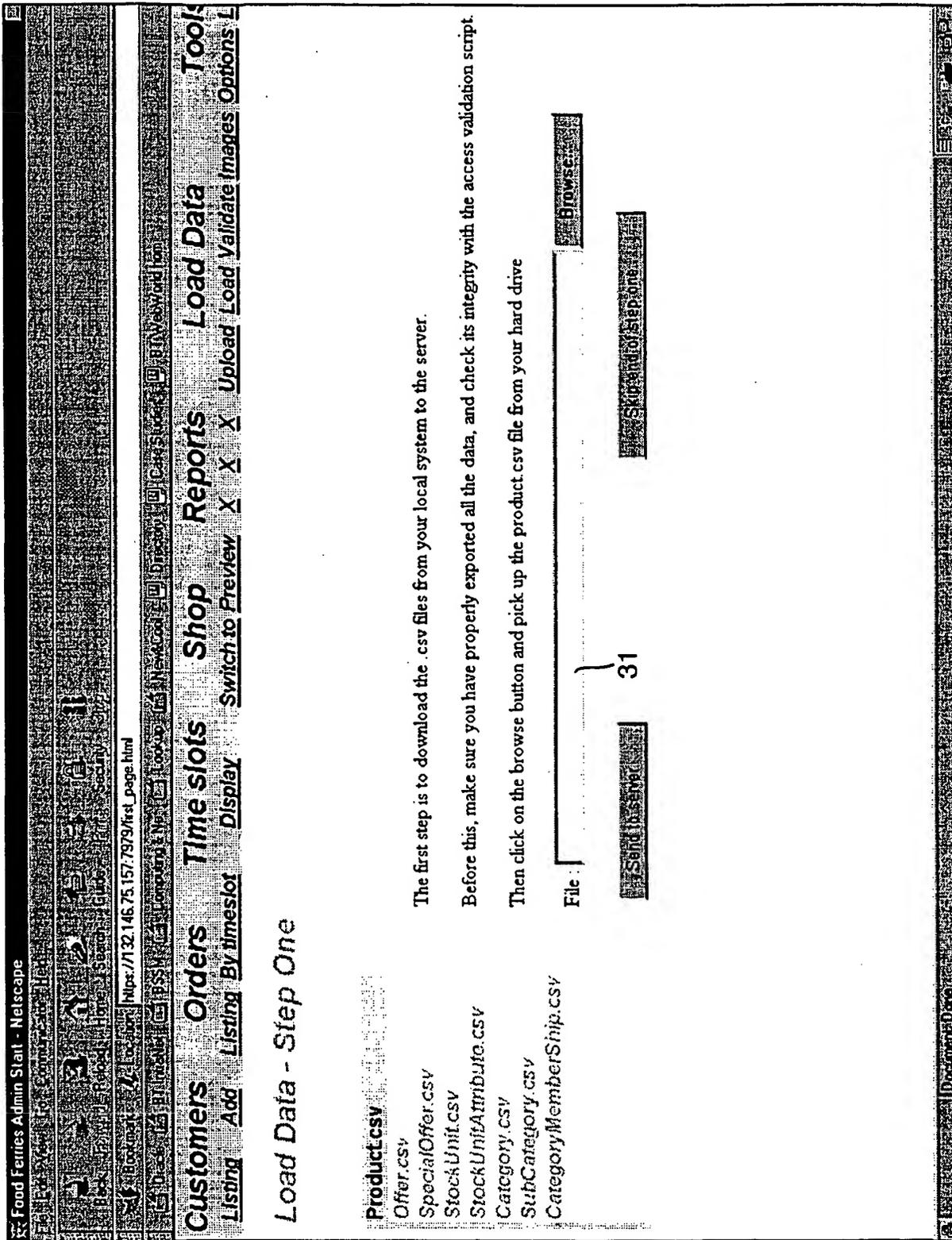
230

Alcohol
 Baby Bits
 Catering
 Stationery and
 Sundries
 Fresh Produce
 Frozen Foods
 Hot Beverages
 Laundry and
 Cleaning
 PUT WITH
 SAUCES
 Pet Products
 Pizza Bases

Clear
 Load
 Save
 Order

My List

Fig.3.



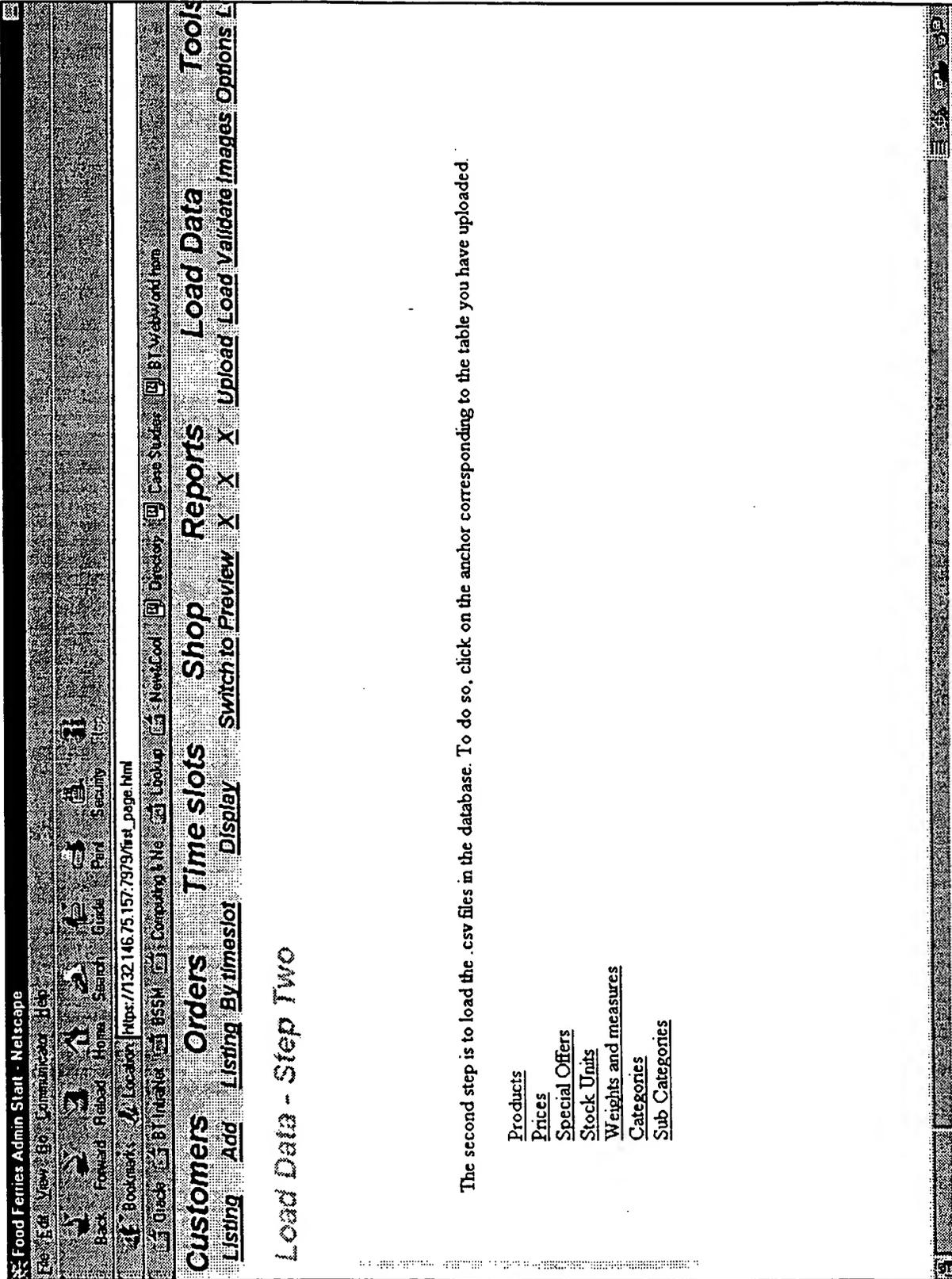


Fig. 4.

The second step is to load the .csv files in the database. To do so, click on the anchor corresponding to the table you have uploaded.

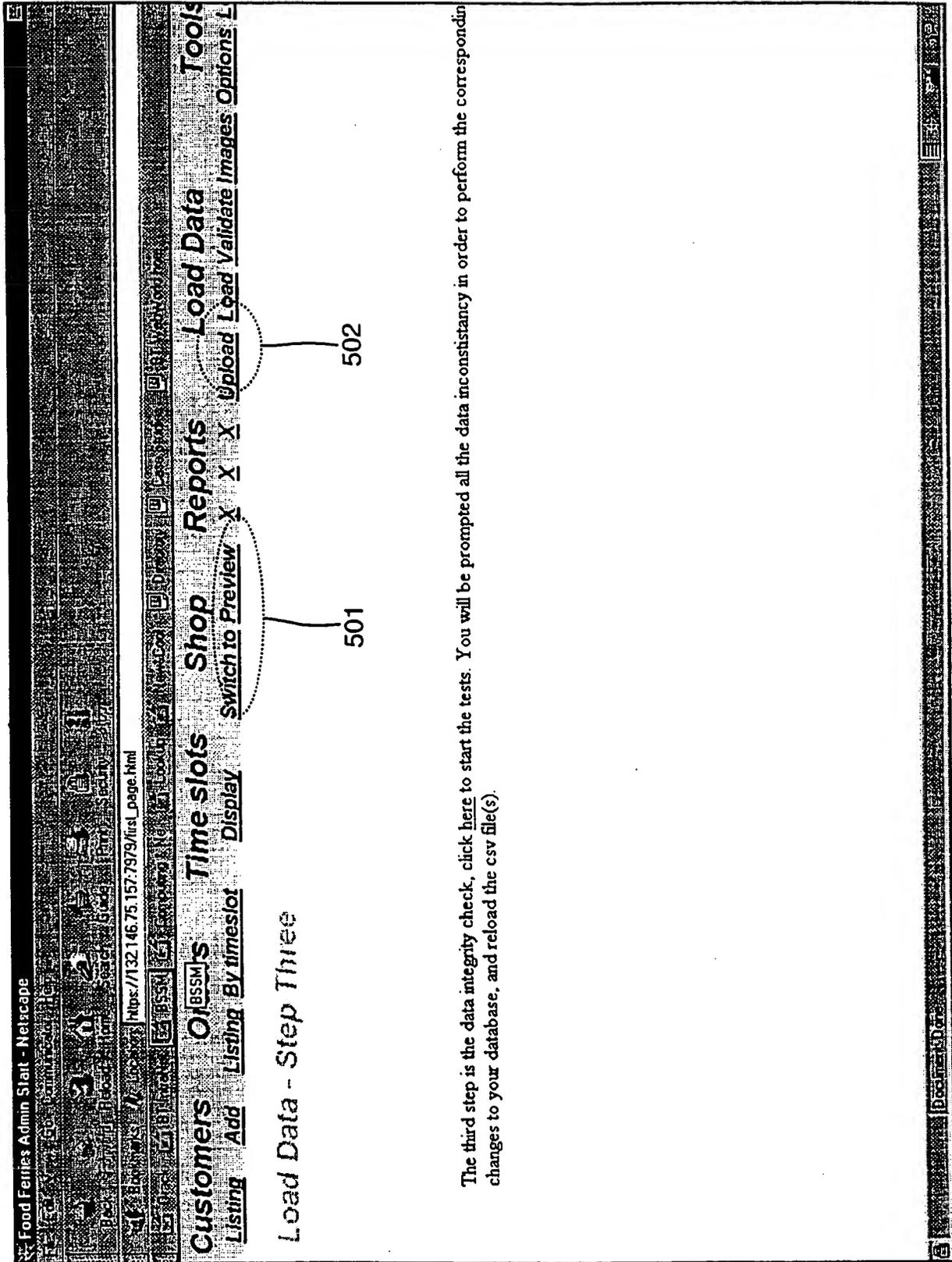


Fig.5.

Food Ferris Admin Start - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Guide Print Security Help

Bookmarks Location: http://132.146.75.157:7979/first_page.htm

Open Close Browsers BSSM Copying No Lookup NewLoad Directory Case Studies BT Web/and hom

Customers **Orders** **Time slots** **Shop** **Reports** **Load Data** **Tools**

[Listing](#) [Add](#) [Listing](#) [By timeslot](#) [Display](#) [Switch to Preview](#) [X](#) [X](#) [X](#) [Upload](#) [Load](#) [Validate](#) [Images](#) [Options](#) [L](#)

SQL*Loader: Release 7.3.4.0.1 - Production on Thu Apr 16 11:21:11 1998

Copyright (c) Oracle Corporation 1979, 1996. All rights reserved.

Control File: /oracle7/app/oracle/foodferry/staging/product.ct1
 Data File: /oracle7/app/oracle/foodferry/tmp/product.csv
 Bad File: /oracle7/app/oracle/foodferry/staging/product.bad
 Discard File: none specified

(Allow all discards)

Number to load: ALL
 Number to skip: 0
 Errors allowed: 50
 Bind array: 64 rows, maximum of 65536 bytes
 Continuation: none specified
 Path used: Conventional

Table PRODUCT, loaded from every logical record.
 Insert option in effect for this table: REPLACE

| Column Name | Position | Len | Term | Encl | Datatype |
|----------------------|----------|-----|------|------|---------------|
| PRODUCT_NO | FIRST | * | , | O(*) | CHARACTER |
| PRODUCT_NAME | NEXT | * | , | O(*) | CHARACTER |
| PRODUCT_DESC | NEXT | * | , | O(*) | CHARACTER |
| MANUFACTURER | NEXT | * | , | O(*) | CHARACTER |
| PRODUCT_TYPE | NEXT | * | , | O(*) | CHARACTER |
| SHIPPING_CHARGE | NEXT | * | , | O(*) | CHARACTER |
| EFFECTIVE_START_DATE | NEXT | * | , | O(*) | DATE DD-MM-YY |

Fig.6.

Fig.7.

